機能 → 構造 → 振舞 → 要素技術 → 独自性

プロジェクト VDM

要求からのアーキテクチャ導出

要求から機能を分析するにあたり、ETロボコン2009～2011までの競技規約から機能要求を抽出し、Feature図にまとめた。

「難所」類はオプションとしたが、好成績を得るためにはなるべく多くの難所をクリアする必要がある。

Feature図から分かったこと

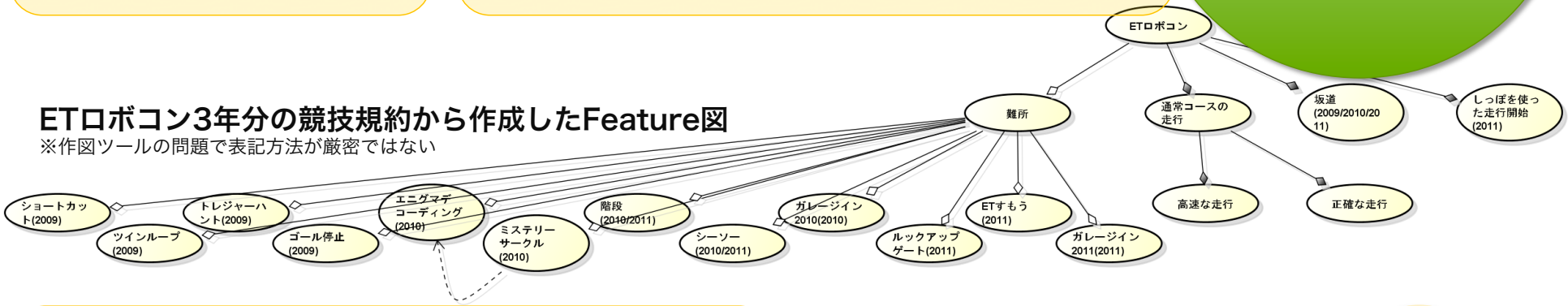
- **共通性**
  通常コースを高速かつ正確に走行、坂道の走破、尻尾を使った走行開始
- **可変性**
  各難所に対応した個別の走行機能

オプションとしての個々の難所の選択

大会毎の競技規約、競技参加者の意思によって幅がある
→SPL「的」な考え方ができる？ ※ SPL = ソフトウェアプロダクトライン

ETロボコン3年分の競技規約から作成したFeature図

※作図ツールの問題で表記方法が厳密ではない



SPL「的」な考え方

具体的に複数の製品が想定されてはいないが、明確な可変性に関する指針を持ったアーキテクチャを選択することで、バリエーション作りは容易になる

複数年対応

大会毎に変わる難所への対応を部品化し、部品の交換で複数年対応が可能となる

難所攻略実装の段階化

難所への対応を部品化し、段階的に追加していけるようにすれば、限られたリソースで可能な範囲の難所攻略実装を、高効率、並行的に進めることが可能となる

Feature図を手がかりに、共通性と可変性について、直交性を検討すると、オプション部分が必須部分に依存していないことが分かる。共通で必要な機能も、個別の難所攻略実装と同列に扱えると分かる。

そこでSPL「的」な考えに基づいた変更容易性を、アーキテクチャにどう持たせるかを検討した結果、設定された走行区間で部品化された走行戦略を切り替える「区間切替」アーキテクチャの採用を決めた。

区間毎に部品化し、部品間の依存を排することで、複雑さを抑え、システムの堅牢性を担保できる。

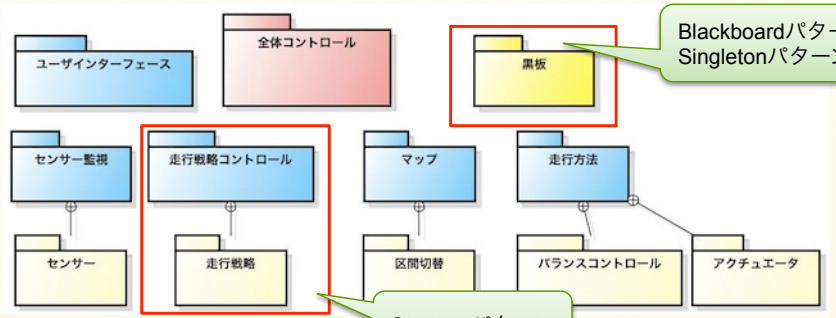
プロジェクトVDM

前ページでの要求分析にある「変更容易性」と、当チームのコンセプトである「堅牢性」を兼ね備えたアーキテクチャ構造を作成した。
特徴としては、Strategyパターンを用いた走行戦略重視の構造と、Blackboardパターンを用いたデータの一貫性である。

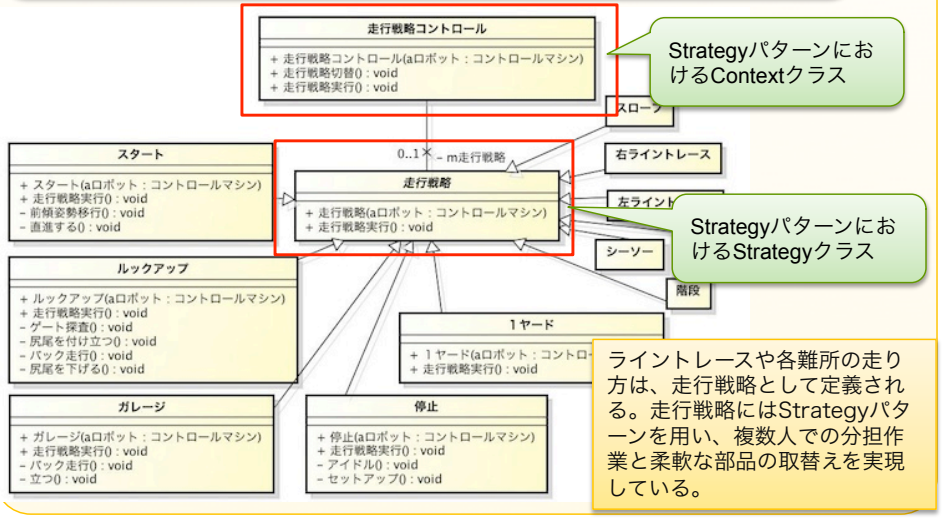
本ページのクラス図は、全てVDMモデルから自動生成したものである。
そのため、操作や属性の定義は、VDM++の文法に準拠している。
また、名前付けには以下のルールがある。

- 操作の仮引数名：a～
- Getter, Setter操作名：get～, set～
- 属性名：m～
- インスタンス生成操作名：create～
- true, false判定操作名：is～

パッケージレベルの構造



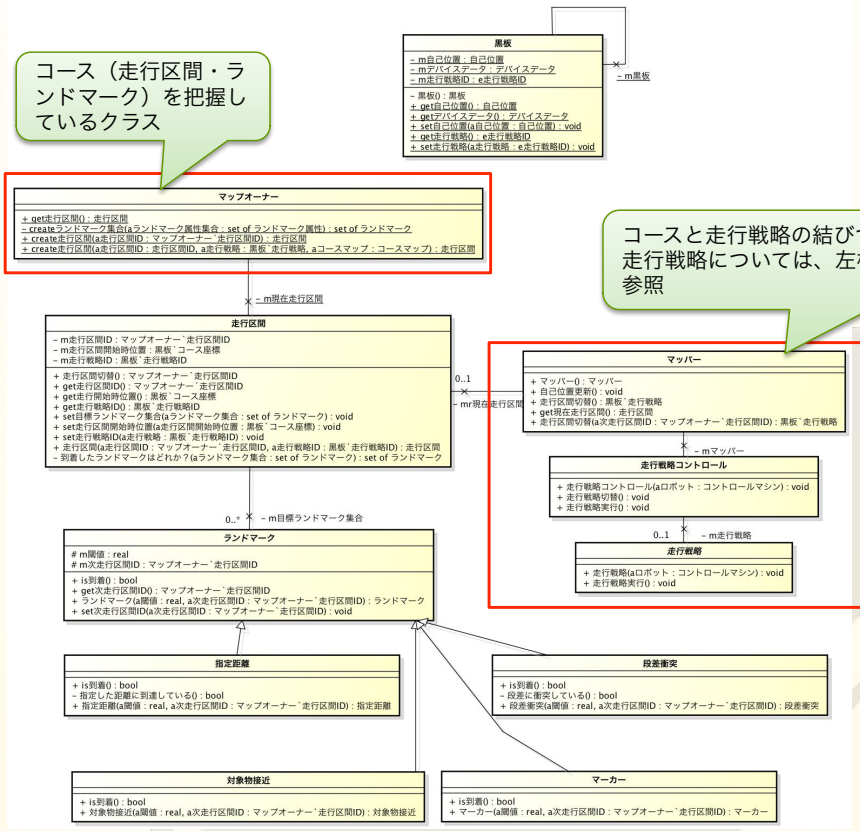
走行戦略の構造（Strategyパターンの適用）



ライントレースや各難所の走り方は、走行戦略として定義される。走行戦略にはStrategyパターンを用い、複数人での分担作業と柔軟な部品の取替えを実現している。

区間切替の構造（区間切替関連のみの抜粋）

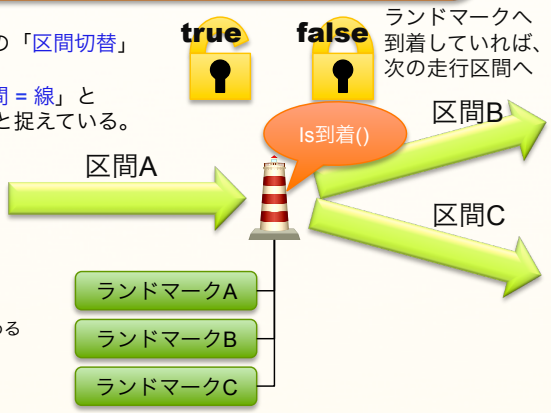
区間切替の仕組みについては、次ページを参照



プロジェクト VDM

区間切替の基本システムと切替時の振る舞い

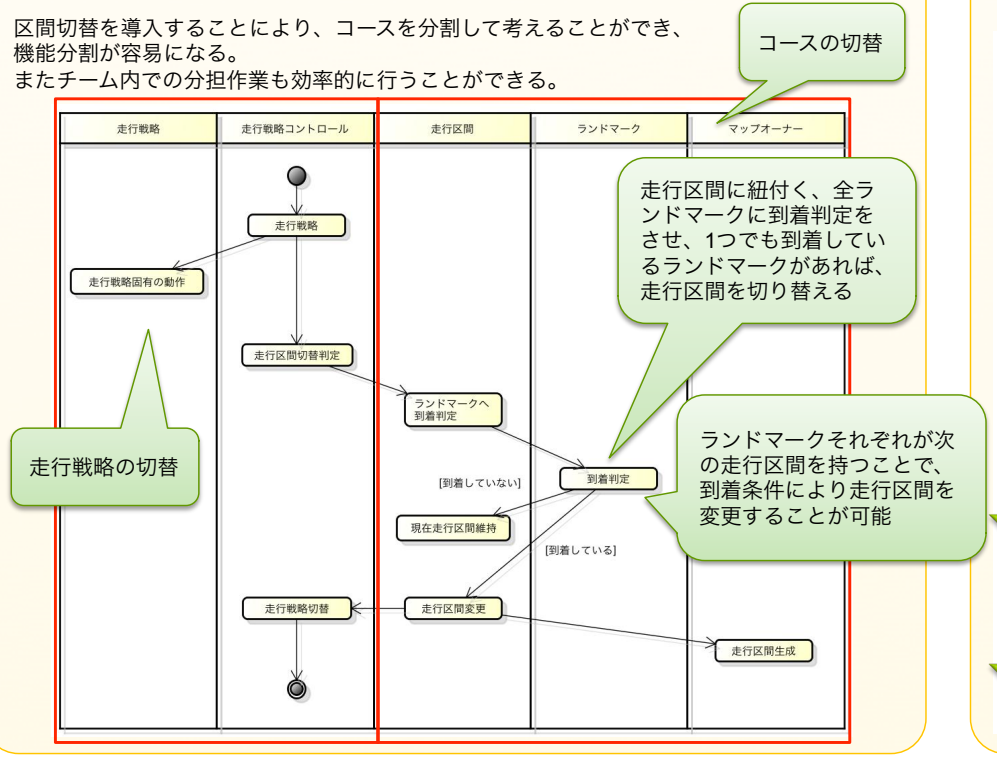
アーキテクチャの要となるのが、この「区間切替」である。
これは、コースのレイアウトを「区間 = 線」と「ランドマーク = 点」の組み合わせと捉えている。



各区間は、以下の構成から成る。

- **現在走行区間**
  現在走行している区間
- **ランドマーク集合**
  現在の走行区間に紐付く目標地点
  ランドマークによって、次走行区間が変わる
- **次走行区間**
  次に走行する区間

区間切替を導入することにより、コースを分割して考えることができ、機能分割が容易になる。
またチーム内での分担作業も効率的に行うことができる。



区間切替における省メモリ作戦

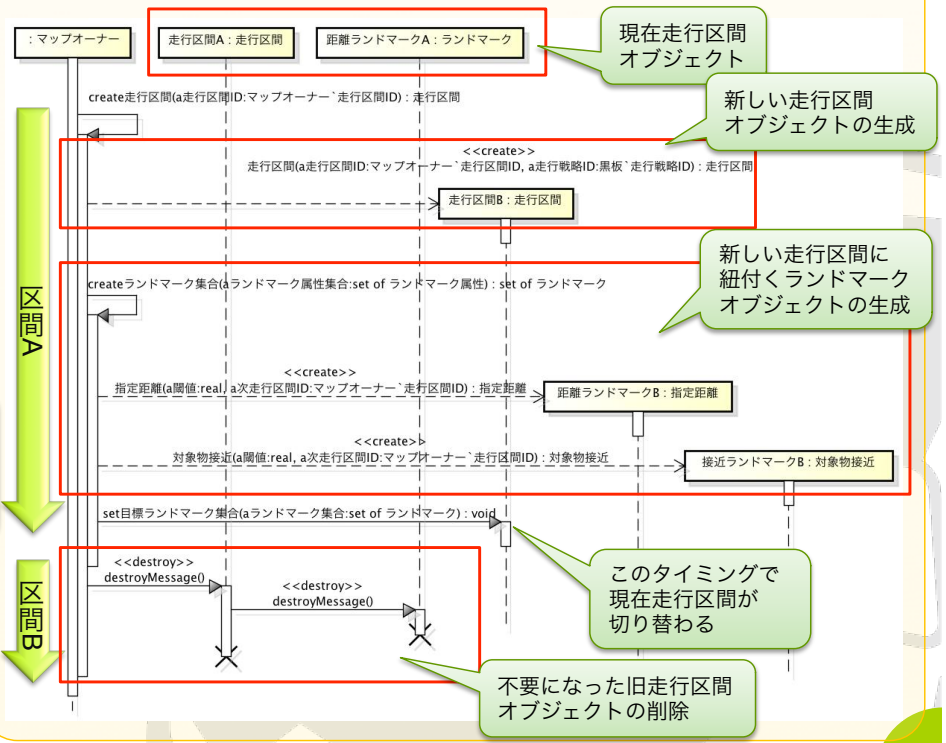
区間とランドマーク、これらを全て合わせると、論理的なコースそのものとなる。
当然、全区間＆全ランドマークのオブジェクトを保持していれば、何も考えずに現在区間を切り替えるだけで済む。

しかし、組込みという特性上、使用できるメモリ空間には限りがあり、それは豊富ではない。
そこで、最小限の組み合わせのみを常に保持することで、できる限りメモリ使用量を抑えようとしたのが、この省メモリ作戦である。

保持するオブジェクト

- 現在走行区間（走行区間クラスのオブジェクト）
- 現在走行区間に紐付くランドマークの集合（ランドマーククラスのオブジェクト群）
- 次走行区間のID（オブジェクトではなくIDのみ）

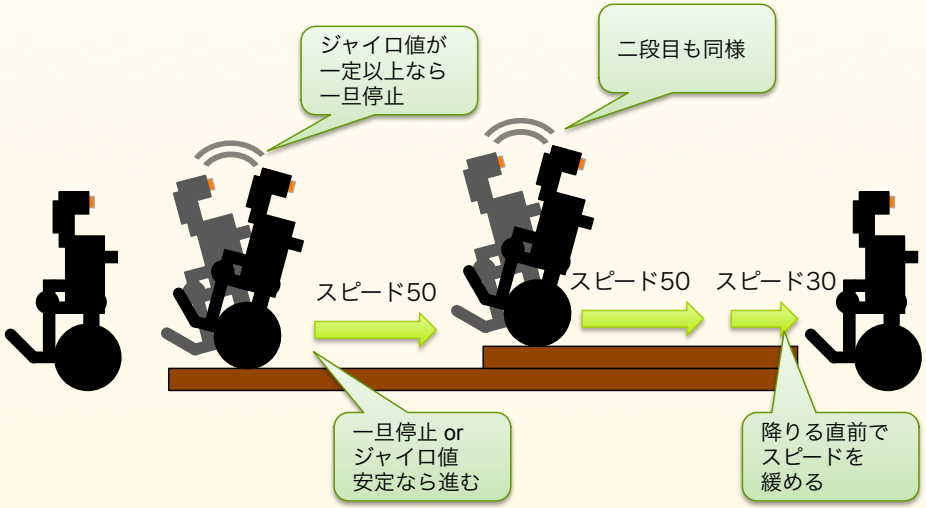
この仕組は、マップオーナーという論理的なコースを司るクラスが制御している。以下のシーケンス図は、この区間切替時のオブジェクト生成と削除のタイミングを表している。



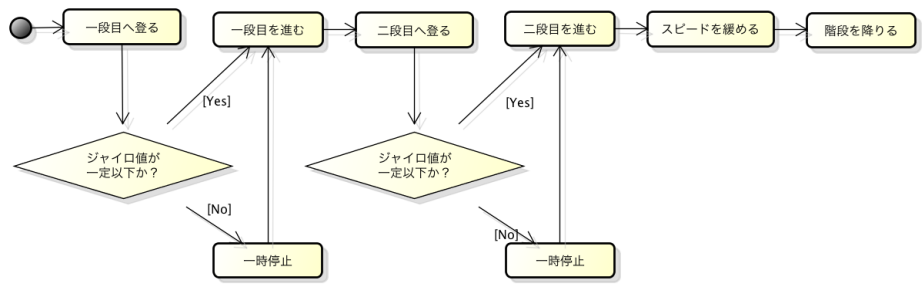
階段攻略は実装者の性格が出た慎重な走り

階段の攻略は、実装を担当したメンバーの性格が良く出ている。
1段ごとに自らの状態を観察し、もしもバランスを崩している場合は一旦立ち止まる。
これにより、慎重かつ安定した階段の攻略が可能となっている。

慎重な走りは自分の足元を固めることにある！



フローチャートで表現すると



ルックアップゲートへ向かうための姿勢作り

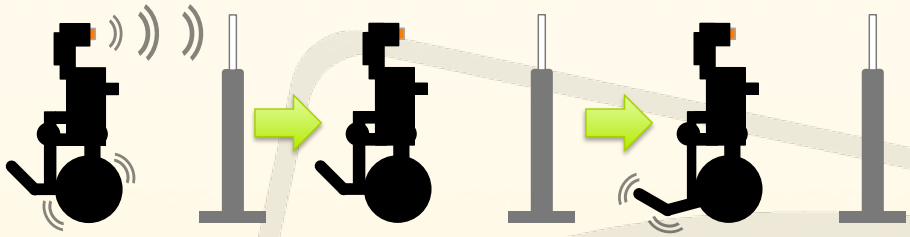
ルックアップゲートは、NXTを傾けた上で通過しなければ、頭部がゲートに接触する。
そのため、ルックアップゲートへ向かうための後傾姿勢作りは重要である。

当チームでは、その姿勢作りに一工夫加えている。
それは、停止した状態から後傾姿勢を作る手段として、少しだけバックすることである。

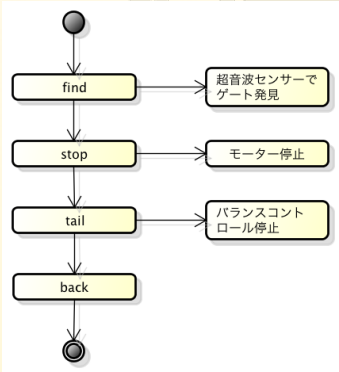
これにより、安定かつ自然な後傾姿勢作りが可能となる。

find → stop → tail → back

ゲートを発見　一時停止　尻尾を降ろす



アクティビティ図で表現すると



バックで姿勢作り

プロジェクト VDM

形式手法VDMによるアーキテクチャの事前検証

組込みシステムは検証が難しい。

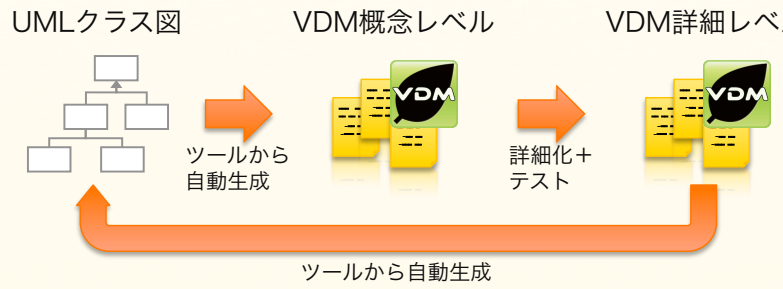
設計をし、ソースコードを書き、実機にモジュールを載せなければ動作確認ができないことが多いからだ。

この点は、モデルを記述することでいくらか軽減することができる。しかし、例えばUMLでアーキテクチャの構造と振る舞いを記述しただけでは、そのアーキテクチャが想定通り動作するかを確認することは難しい。

当チームでは、アーキテクチャモデルレベルでの実現性確認と動作検証をするために、形式手法であるVDMを用いた。VDMでは、記述したモデルを動作させてテストすることができるため、アーキテクチャの事前検証に向いている。

今回は、前述した区間切替部分にVDMを適用し、アーキテクチャの実現性を見た上で実装をした。これにより、事前にアーキテクチャの問題点を洗い出し、実現性があることを確認することができた。そのため、アーキテクチャを重視した当チームの開発は、非常に効率的に行うことが可能となった。

VDMを用いたアーキテクチャ検証イメージ



実現したいアーキテクチャの構造をクラス図で表現する

実現性と動作検証をするために、UMLからVDMの概念レベルを自動生成

VDMモデルを詳細化＋テストケース記述、テストを経た結果をクラス図へ自動フィードバック

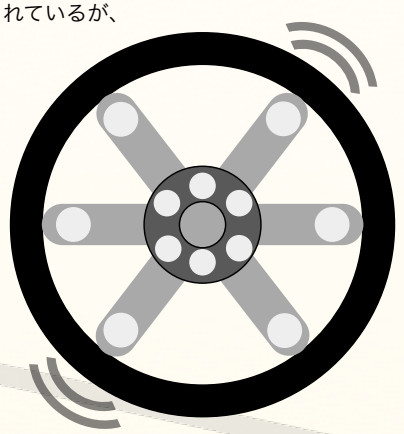
車輪は走るだけではない！UIでもあるのだ！

試走会を始め、走らせながら色々な試行をすることは良くある。
特に、環境に合わせてPID制御のパラメータを変更したり、難所をピンポイントで試すような時に、毎回ビルドして転送していたのでは、手間がかかる。

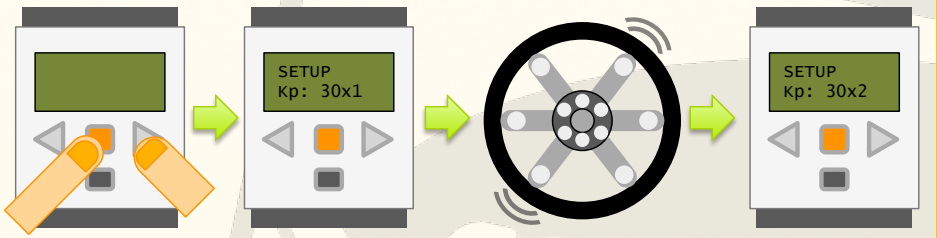
今年からBluetoothによるPCとのやり取りが解禁されているが、Bluetooth不調時の対策も必要である。そこで当チームでは、車輪とモーターのエンコーダ値に目をつけた。

車輪を回すことをダイヤルとして捉え、LCDとボタンと車輪を用いて、各種パラメータの調整や走行戦略選択実行を容易とした。

これにより、PCを用いずにNXT本体のみで様々な試行をすることが容易となった。



操作方法



オレンジボタンと左ボタンを同時に押すことでセットアップへ

セットアップモードにて、どの項目を変更するか選択

PID制御パラメータのような値は車輪回転で1メモリずつ変更可

セットアップ完了後、再び走らせる

変更可能項目
・PID制御パラメータ（Kp, Ki, Kd, 速度），コース選択（in / out），走行戦略選択